MonoGame on Windows Phone 8 Post-Processing Your Game

This article explains how to compile and use shaders with MonoGame on Windows Phone 8 for post-processing.

10 Mar 2013

Introduction

When you develop a game at some point you start thinking about adding special effects to your graphics pipeline. Doing this is beneficial for several reasons

- For having effects that cannot be created by an artist (refraction, reflection, water effects)
- For setting additional mood for a gameplay (bloom, blur, filter effects)
- For making your games just look cooler (all sorts of distortion effects)

NOTE: this is not a "getting started" tutorial with MonoGame. For information on where to get started please follow this article XNA Games On Windows Phone 8 with Monogame.

Let's Start

First of all we need to create a new MonoGame WP8 project, add `MonoGame.WindowsPhone` as an existing project, reference all needed libraries like SharpDX in MonoGame project (if these are not already referenced) build it and we are ready to go!.

Landscape Mode

The majority of games I've seen on mobile devices are presented in landscape. For some reasons when using MonoGame we don't have a landscape mode just out of the box, so we will have to deal with it on our own. Luckily that's pretty easy. There's even a long discussion thread concerning landscape modes at http://monogame.codeplex.com, however I think it will be helpful to have the solution here.

We will need a render target that will be used for all of the drawing stuff and at the very end of Draw function will be presented to the screen with rotation of 90 degrees and position offset. For this we declare a class level variable

```
RenderTargtet2D finalImageTarget;
```

and initialize it in LoadContent method as

```
finalImageTarget = new RenderTarget2D(GraphicsDevice, 800, 480);
```

NOTE: in this article I will be using WVGA resolution only for the sake of simplicity. That's pretty easy to get current resolution of device. Have a look here for multiple resolutions on WP8

Locate the

```
Draw(GameTime gameTime)
```

function and replace it's content with

```
GraphicsDevice.SetRenderTarget(finalImageTarget);
GraphicsDevice.Clear(Color.CornflowerBlue);

// we will do all of our drawing here

GraphicsDevice.SetRenderTarget(null);
spriteBatch.Begin();
spriteBatch.Draw(finalImageTarget, new Vector2(480, 0), null, Color.White,
MathHelper.PiOver2, Vector2.Zero, Vector2.One, SpriteEffects.None, 0f);
spriteBatch.End();
```

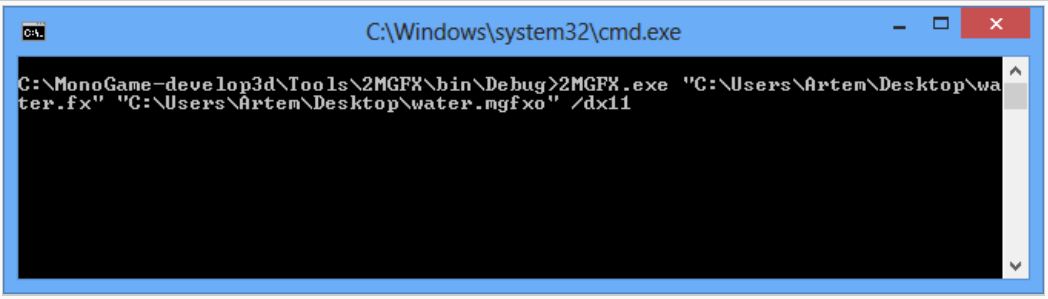

(C) Copyright Nokia 2014. All rights reserved.

```
base.Draw(gameTime);
```

Compiling Shaders for Windows Phone 8

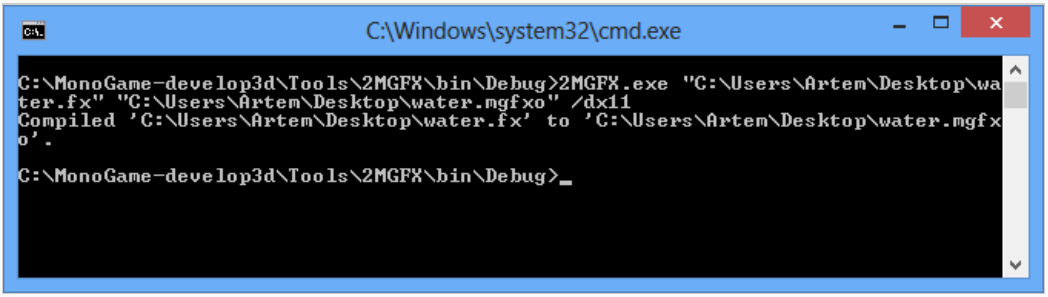
There are two ways to include shaders to your project. The first one is compilation with 2MGFX tool and adding the output as embedded resource. The second one is using the `MonoGame.ContentPipeline`. We will stick to the first one in this article.

So navigate to your MonoGame-develop3d folder, locate Tools folder and 2MGFX folder inside it. Open the project in Visual Studio and build it. After a successful build, open project's output folder and start command line from that folder. Follow the syntax shown next: **2MGFX.exe [source_file_with.fx] [destination_file.mgfxo] /dx11**. The /dx11 flag is used to specify the platform the shader should be compiled for. in my case it looks like this:



Compiling shaders with 2MGFX tool

If compilation is successful you will see something like this



Compiling shaders with 2MGFX tool - success

There's one thing I have to mention. In this article I'll be describing pixel shaders only. There are two things you should remember when writing pixel shaders for WP8 - the signature of `PixelsShader` function **must** include the POSITION. This is required not because of MonoGame or SharpDX but because of preserving alignment for other inputs. It's the HLSL thing.

So having something like this

```
float4 Water(float4 pos: POSITION, float4 col: COLOR, float2 coords: TEXCOORD0) : COLOR0
```

as a PixelShader function declaration will always exclude any errors of declarations or weird behaviors.

And the second one is the compilation directive which include the DirectX feature level. For Windows Phone 8 that's feature level 9.3.

```
technique WaterTech
{
    pass P0
    {
        PixelShader = compile ps_4_0_level_9_3 Water();
    }
}
```

Now when we are locked and loaded we are ready to post-process.

(C) Copyright Nokia 2014. All rights reserved.

2D Water Effect

We will mix some rendertargets and UV coordinates offsets to create waves. The basic flow is next:

- set our mainTarget as rendertarget, draw the image to this rendertarget
- set water target and draw mainTarget using water shader to it
- set finalImageTarget and draw first mainTarget at position Vector.Zero and then draw water target at position Vector2(0, waterLevel)
- set null as RenderTarget and present finalImageTarget to screen with rotation and position offset.

Please see the code and the shader in 2DWaterEffect Project for detailed information. And here's the result

Water Effect in action

RGB Split Effect

Have you ever played Dirt 2 or Dirt 3? There's a pretty nice effect in cut scenes over there when Camera shakes. It looks like the colors are being split into separate channels and shifted in different directions for a short period of time Let's try replicating the same effect for post-processing our WP8 games.

For this effect we will use random value for distortion and shifting. Also this effect is much simpler than the water effect, so we will use less resources in code. We need the actual shader, only one rendertarget for final rendering and a random number generator. If TouchPanel contains any values we will reload our `DisplacementScroll` variable with random values and pass to the shader for drawing.

Here go the results:

No Distortion

(C) Copyright Nokia 2014. All rights reserved.

RGB Split Effect

Another split Effect

However this effect looks much better in action, see it yourself: The media player is loading...

In addition, imagine some heavy rock music playing in background and this effect taking place in menu of a game. That would be pretty cool.

Please find the code and shader in RGBSplitEffect project inside the solution

Summary

We've seen some techniques for using pixel shaders with MonoGame on WP8. We've seen how we can develop and compile shaders for using with MonoGame These small GPU programs can greatly improve the overall image quality of your games and present the game in correct mood to the player.

The Code

There you go! Try it, experiment with it File:2DWaterEffect.zip

(C) Copyright Nokia 2014. All rights reserved.

(C) Copyright Nokia 2014. All rights reserved.